Amendments to the Claims

This listing of claims will replace all prior version, and listings, of claims in the application.

Listing of Claims:

1. (currently amended) An operand file comprising:

at least one pair of future state and architecture state pointers;

an operand queue including at least one entry; and

a reference counter associated with each operand queue entry, whose count indicates the number of registers mapped to the associated operand queue entry.

2. (original) The operand file of claim 1, in which a free operand queue entry is assigned to hold a future value of a register of an instruction by writing the free entry's number into the register's future state pointer and incrementing the free entry's reference point.

3. (original) The operand file of claim 2, in which the assigned entry number is written to the register's architectural state pointer and the reference count of the entry previously assigned to the register is decremented upon completion of the instruction.

4. (original) The operand file of claim 3, in which each register is assigned a unique operand queue entry upon a reset.

5. (original) The operand file of claim 3, in which all registers that have undefined value upon reset are assigned to at least one operand queue entry and each of the registers

that have defined value upon reset is assigned a unique entry upon a reset.

6. (original) The operand file of claim 3, in which the entry number previously assigned to the register is obtained from the register's future state pointer.

7. (original) The operand file of claim 3, in which the entry number previously assigned to the register is obtained from the register's architectural state pointer.

8. (original) The operand file of claim 3, in which each of the architectural state pointer is copied to its corresponding future state pointer when processing an exception condition.

9. (original) The operand file of claim 3, in which a cancelled instruction does not modify associated architectural state pointers but the reference count of the entry assigned to the register is documented.

10. (original) The operand file of claim 3, in which a register-copy instruction is executed by copying the operand queue entry number in a source register's future state pointer to a destination register's future state pointer and incrementing the reference count of the associated entry.

11. (original) The operand file of claim 10, in which a register-copy instruction is completed by copying the operand queue entry number in the source register's architectural state pointer to the destination register's architectural state pointer and decrementing the reference count of the entry previously assigned to the destination register.

12. (original) The operand file of claim 10, in which a register-copy instruction is completed by reading the operand queue entry number in the destination register's future state pointer at decode time and writing the entry number to the destination register's architectural state pointer and decrementing the reference count of the entry previously assigned to the destination register.

13. (original) The operand file of claim 10, in which a register-copy instruction copies the future value of the source register to the operand queue entry assigned to the destination register when the reference count of the entry in the source register's future state pointer is at its maximum value.

14. (original) The operand file of claim 10, in which a register-copy instruction copies the future value of the source register to the operand query entry assigned to the destination register when the reference count of any entry is at its maximum value.

15. (original) The operand file of claim 1, in which an immediate operand is assigned a free operand query entry by incrementing the reference count of the free entry.

16. (original) The operand file of claim 15, in which the immediate operand is written to the operand queue at any time before the associated instruction needs to read the operand file.

17. (original) The operand file of claim 16, in which the entry assigned to the

immediate operand is decremented when the associated instruction is completed.

18. (original) The operand file of claim 16, in which a cancelled instruction with an immediate operand does not modify associated architectural state pointers but the reference count of the entry assigned to hold the immediate operand is decremented.

19. (original) The operand file of claim 16, in which the entry assigned to the immediate operand is decremented as soon as the immediate operand is read.

20. (original) The operand file of claim 3 in which each thread in a multithreaded processor has its own set of architectural and future state pointers but shares one operand queue.

21. (original) The operand file of claim 20 in which all registers that have undefined values in a thread is assigned to at least one free operand queue entry by writing the at least one free entry's number into the thread's architectural and future state pointers and incrementing the at least one entry's reference count by the number of registers.

22. (original) The operand file of claim 20 in which a register in a first thread is copied to a register in a second thread by copying the operand queue entry number in the architectural state pointer of the register in the first thread to the architectural and future state pointers of the register in the second thread and incrementing the reference count of the associated operand queue entry.

23. (original) The operand file of claim 20 in which a register in a first thread is

copied to a register in a second thread by copying the operand queue entry number in the future state pointer of the register in the first thread to the architectural and future state pointers of the register in the second thread and incrementing the reference count of the associated operand queue entry.

24. (original) The operand file of claim 20 in which the reference count of the entry in each of a thread's architectural state pointer is decremented by 1 upon terminating the thread.

25. (currently amended) A computer adapted to include an operand file, the operand file comprising:

at least one pair of future state and architecture state pointers;

an operand queue including at least one entry; and

a reference counter associated with each operand queue entry, whose count indicates the number of registers mapped to the associated operand queue entry.

26. (original) The computer of claim 25, in which all registers that have undefined value upon reset are assigned to at least one operand queue entry and each of the registers that have defined value upon reset is assigned a unique entry upon a reset.

27. (original) The computer of claim 25, in which each register is assigned a unique operand queue entry upon a reset.

28. (original) The computer of claim 25, in which a free operand queue entry is

assigned to hold a future value of a register of an instruction by writing the free entry's number into the register's future state pointer and incrementing the free entry's reference count.

29. (original) The computer of claim 28, in which the assigned entry number is written to the register's architectural state pointer and the reference count of the entry previously assigned to the register is decremented upon completion of the instruction.

30. (New) An operand file comprising:

at least one pair of future state and architecture state pointers;

an operand queue including at least one entry; and

a reference counter associated with each operand queue entry, in which a free operand queue entry is assigned to hold a future value of a register of an instruction by writing the free entry's number into the register's future state pointer and incrementing the free entry's reference point, in which the assigned entry number is written to the register's architectural state pointer and the reference count of the entry previously assigned to the register is decremented upon completion of the instruction, and in which a register-copy instruction is executed by copying the operand queue entry number in a source register's future state pointer to a destination register's future state pointer and incrementing the reference count of the associated entry.